

# A Characterization Theorem and An Algorithm for A Convex Hull Problem

Bahman Kalantari

Department of Computer Science, Rutgers University, NJ  
kalantari@cs.rutgers.edu

## Abstract

Given a set  $S = \{v_1, \dots, v_n\} \subset \mathbb{R}^m$  and a point  $p \in \mathbb{R}^m$ , testing if  $p \in \text{conv}(S)$ , the convex hull of  $S$ , is a fundamental problem in computational geometry and linear programming. First, we prove a Euclidean *distance duality*: Either for each  $p' \in \text{conv}(S) \setminus \{p\}$ ,  $\exists v_j \in S$  such that  $d(p', v_j) \geq d(p, v_j)$ ; or  $\exists p' \in \text{conv}(S)$  such that  $d(p', v_i) < d(p, v_i), \forall i$ . The former case holds if and only if  $p \in \text{conv}(S)$  and the latter if and only if  $p \notin \text{conv}(S)$ . Utilizing this duality, we describe a simple fully polynomial time approximation scheme, called the *Triangle Algorithm*: Given  $\epsilon > 0$ , in  $O(mn\epsilon^{-2})$  arithmetic operations it computes  $p' \in \text{conv}(S)$  where, either  $d(p', p) \leq \epsilon d(p, v_j)$  for some  $j$ ; or  $d(p', v_i) < d(p, v_i), \forall i$ . When  $p \notin \text{conv}(S)$ , we show  $d(p, p')$  estimates the distance from  $p$  to  $\text{conv}(S)$  to within a factor of two. We also show how to solve general LP via the Triangle Algorithm and give a corresponding complexity analysis. In particular, we prove a sensitivity theorem that converts LP feasibility with bounded domain into a convex hull problem, then gives the necessary accuracy for computing an  $\epsilon$ -approximate solution. We also contrast the theoretical performance of the Triangle Algorithm with the *sparse greedy approximation* (equivalent to Frank-Wolfe and Gilbert algorithms) for the minimization of a convex quadratic over a simplex, a problem arising in machine learning, approximation theory, and statistics.

**Keywords:** Convex Hull, Linear Programming, Duality, Approximation Algorithms, Sparse Greedy Approximation, Frank-Wolfe Algorithm, Support Vector Machines.

## 1 Introduction

Given a set  $S = \{v_1, \dots, v_n\} \subset \mathbb{R}^m$  and a distinguished point  $p \in \mathbb{R}^m$ , we consider the problem of testing if  $p$  lies in  $\text{conv}(S)$ , the convex hull of  $S$ . Throughout the article we shall refer to this problem as the *convex hull decision problem*, or simply as problem (P). This is a basic problem in computational geometry and a very special case of the *convex hull problem*, a problem that according to Goodman and O'Rourke [17], is a “catch-all phrase for computing various descriptions of a polytope that is either specified as the convex hull of a finite point set or the intersection of a finite number of halfspaces.” The descriptions include those of vertices, facets, and adjacencies.

Problem (P) is not only a fundamental problem in computational geometry, but in linear programming (LP). This can be argued on different grounds. On the one hand problem (P) is a very special case of LP. On the other hand, it is well known that through LP duality theory the general LP problem may be cast as a single LP feasibility problem, see e.g. Chvátal [7]. The LP feasibility problem can then be converted into problem (P) via several different approaches. To argue the significance of (P) in linear programming, it can be justified that the two most famous polynomial-time LP algorithms, the ellipsoid algorithm of Khachiyan [27] and the projective algorithm of Karmarkar [25], are in fact explicitly or implicitly designed to solve a case of problem (P) where  $p = 0$ , see [19]. Furthermore, using an approach suggested by Chvátal, in [19] it is shown that there is a direct connection between a general LP feasibility and this homogeneous case of problem (P), over integer or real input data. For integer inputs all known polynomial-time LP algorithms - when applied to solve problem (P) - would exhibit a theoretical complexity that is polynomial in  $m, n$ ,

and the size of encoding of the input data, often denoted by  $L$ , see e.g. [27]. The number  $L$  can generally be taken to be dependent on the logarithm of  $mn$  and of the logarithm of the absolute value of the largest entry in the input data. These are sometimes known as *weakly polynomial-time* algorithms. No *strongly polynomial-time* algorithm is known for LP, i.e. an algorithm that would in particular solve problem (P) in time complexity polynomial in  $m$  and  $n$ .

Problem (P) finds applications in computational geometry itself, in particular among the variations of the convex hull problem. One such a variation is the *irredundancy problem*, the problem of computing all the vertices of  $\text{conv}(S)$ , see [17]. Clearly, any algorithm for LP can be used to solve the irredundancy problem by solving a sequence of  $O(n)$  convex hull decision problems. Clarkson [9], has given a more efficient algorithm than the straightforward approach of solving  $n$  linear programs. The complexity of his algorithm depends on the number of vertices of the convex hull. Furthermore, by using an approach originally due to Matoušek [31] for solving closely related linear programming problems, Chan [5] has given a faster algorithm for the irredundancy problem. What is generally considered to be the convex hull problem is a problem more complicated than (P) and the irredundancy problem, requiring the description of  $\text{conv}(S)$  in terms of its vertices, facets and adjacencies, see Chazelle [6] who offers an optimal algorithm.

Problem (P) can also be formulated as the minimization of a convex quadratic function over a simplex. This particular convex program has found applications in statistics, approximation theory, and machine learning, see e.g. Clarkson [10] and Zhang [39] who consider the analysis of a greedy algorithm for the more general problem of minimizing certain convex functions over a simplex (equivalently, maximizing concave functions over a simplex). The oldest version of such greedy algorithms is Frank-Wolfe algorithm, [14]. Special cases of the problem include support vector machines (SVM), approximating functions as convex combinations of other functions, see e.g. Clarkson [10]. The problem of computing the closest point of the convex hull of a set of points to the origin, known as *polytope distance* is the case of problem (P) where  $p$  is the origin. In some applications the polytope distance refers to the distance between two convex hulls. Gilbert's algorithm [16] for the polytope distance problem is one of the earliest known algorithms for the problem. Gärtner and Jaggi [15] show Gilbert's algorithm coincides with Frank-Wolfe algorithm when applied to the minimization of a convex quadratic function over a simplex. In this case the algorithm is known as *sparse greedy approximation*. For many other results regarding the applications of the minimization of a quadratic function over a simplex, see the bibliographies in [39], [10] and [15]. Clarkson [10] gives a through analysis of Frank-Wolfe and its variations.

Despite all the previous approaches and achievements in solving problem (P), in various formulations and via various algorithms, investigation of the problem will most likely continue in the future. The present article focuses on solving problem (P) directly. In the process it reveal new and interesting geometric properties of this fundamental convex hull problem, including a characterization theorem, leading to a new separating hyperplane theorem for problem (P), a *distance duality* theorem. It then utilizes the distance duality to describe a very simple algorithm called, *Triangle Algorithm*. Next, it utilizes the Triangle Algorithm, together with a sensitivity theorem, to give a new algorithm for the LP feasibility problem when there are no recession directions. It then analyzes the complexity of solving the general LP feasibility via the Triangle Algorithm.

Before formally describing and proving the results in Section 2 - Section 6, in the remaining of this section we give an overview of the results. In 1.1, we describe the Triangle Algorithm and its properties. In 1.2, we describe the geometry of the Triangle Algorithm and two geometric problems that can be associate as problems dual to problem (P). In 1.3, we review the literature on the properties and complexity of a greedy algorithm for the minimization of convex quadratic over a simplex and contrast these with the theoretical performance of the Triangle Algorithm. In 1.4, we describe the application of the Triangle Algorithm for solving an LP feasibility problem having no recession direction. In 1.5, we describe connections between problem (P) and general LP feasibility and LP optimization, as well as reviewing some of the vast literature on LP. In 1.6, we give the outline of the formal results in the remaining sections.

## 1.1 The Triangle Algorithm and Its Properties

Denoting the Euclidean distance between  $u, w \in \mathbb{R}^m$  by  $d(u, w) = \sqrt{\sum_{i=1}^m (u_i - w_i)^2}$ , the Triangle Algorithm takes as input a set of point  $S = \{v_1, \dots, v_n\}$  and a point  $p$  in  $\mathbb{R}^m$ . It consists of iterating two steps:

**Triangle Algorithm** ( $S = \{v_1, \dots, v_n\}, p$ )

- **Step 1.** Given  $p' \in \text{conv}(S) \setminus \{p\}$ , check if there exists  $v_j \in S$  such that  $d(p', v_j) \geq d(p, v_j)$ . If no such  $v_j$  exists, stop,  $p \notin \text{conv}(S)$ .
- **Step 2.** Otherwise, on the line segment joining  $p'$  to  $v_j$  compute the point nearest to  $p$ . Denote this by  $p''$ . Replace  $p'$  with  $p''$ , go to Step 1.

Clearly, if the algorithm does not terminate in Step 1 it will loop indefinitely. We prove: Given  $\epsilon \in (0, 1)$ , the Triangle Algorithm in at most  $48mn\epsilon^{-2} = O(mn\epsilon^{-2})$  arithmetic operations computes a point  $p' \in \text{conv}(S)$  such that either

$$d(p', p) \leq \epsilon d(p, v_j), \quad \text{for some } j, \quad (1)$$

or

$$d(p', v_i) < d(p, v_i), \quad \forall i = 1, \dots, n. \quad (2)$$

**Remark 1.** Note that the constant in the worst-case complexity is an absolute constant and is independent of the input data. This is due to the fact that the Triangle Algorithm seeks to reduce the relative error,  $d(p', p)/d(p, v_j)$ , which is a more sensible measure for the problem than seeking to reduce the absolute error,  $d(p', p)$ . Clearly, approximation to a prescribed absolute error can also be achieved.

**Remark 2.** By squaring the distances we have

$$d(p', v_j) \geq d(p, v_j) \iff d(p', 0)^2 - d(p, 0)^2 \geq 2v_j^T(p' - p). \quad (3)$$

Thus Step 1 does not require taking square-roots. Also, the computation of  $p''$  requires no square-root operations.

We refer to a point  $p'$  satisfying (1) as *iterate* and  $v_j$  as *pivot point*. We refer to a point  $p'$  satisfying (2) as *witness*. This condition holds if and only if  $p \notin \text{conv}(S)$ . This is because in this case we can prove the Voronoi cell of  $p'$  with respect to the two point set  $\{p, p'\}$  contains  $\text{conv}(S)$  (see Figure 1). Equivalently, the orthogonal bisector of the line segment  $pp'$  separates  $p$  from  $\text{conv}(S)$ .

The set  $W_p$  of all such witnesses is the intersection of  $\text{conv}(S)$  and the open balls,  $B_i = \{x \in \mathbb{R}^m : d(x, v_i) < r_i\}$ ,  $i = 1, \dots, n$ .  $W_p$  is a convex open set in the relative interior of  $\text{conv}(S)$  (see Figure 3).

The justification in the name of the algorithm lies in the fact that in each iteration the algorithm searches for a triangle  $\triangle pp'v_j$  where  $v_j \in S$ ,  $p' \in \text{conv}(S) \setminus \{p\}$ , such that  $d(p', v_j) \geq d(p, v_j)$ . Given that such triangle exists, it uses  $v_j$  as a pivot point to “pull” the current iterate  $p'$  closer to  $p$  to get a new iterate  $p'' \in \text{conv}(S)$ .

The correctness of the Triangle Algorithm lies in a new duality (theorem of the alternative) we call *distance duality*:

**Distance Duality**

Precisely one of the two conditions is satisfied:

- (i): For each  $p' \in \text{conv}(S) \setminus \{p\}$ , there exists  $v_j \in S$  such that  $d(p', v_j) \geq d(p, v_j)$ ;
- (ii): There exists  $p' \in \text{conv}(S)$  such that  $d(p', v_i) < d(p, v_i)$ , for all  $i = 1, \dots, n$ .

The first condition is valid if and only if  $p \in \text{conv}(S)$ , and the second condition if and only if  $p \notin \text{conv}(S)$ . From the description of the Triangle Algorithm we see that given a point  $p' \in \text{conv}(S)$  that is not a witness, having  $d(p, p')$  as the current *gap*, the Triangle Algorithm moves to a new point  $p'' \in \text{conv}(S)$  where the new gap  $d(p, p'')$  is reduced. We will prove that when  $p \in \text{conv}(S)$ , the number of iterations  $K_\epsilon$ , needed to get an approximate solution  $p'$  satisfying (1) is bounded above by  $48\epsilon^{-2} = O(\epsilon^{-2})$ . In the worst-case each iteration of Step 1 requires  $O(mn)$  arithmetic operations. However, it may also take only  $O(m)$  operations. The number of arithmetic operations in each iteration of Step 2 is only  $O(m)$ . Thus the Triangle Algorithm is a fully polynomial-time approximation scheme whose complexity for computing an  $\epsilon$ -approximate solution is  $O(mn\epsilon^{-2})$  arithmetic operation. In particular, for fixed  $\epsilon$  the complexity of the algorithm is only  $O(mn)$ .

The worst-case iteration complexity estimate is under the assumption of worst-case performance in each iteration of the algorithm. Hence, in practice a more efficient complexity is to be expected. The algorithm is geometric in nature, simple in description, and very easy to implement.

When  $p \notin \text{conv}(S)$ , the Triangle Algorithm does not attempt to compute the closest point to  $p$ , say  $p_* \in \text{conv}(S)$ , rather a separating hyperplane. However, by virtue of the fact that it finds a special separating hyperplane, i.e. a hyperplane orthogonally bisecting the line  $pp'$ , it in the process computes an approximation to  $d(p, p_*)$  to within a factor of two. More precisely, any witness  $p'$  satisfies the inequality

$$.5d(p, p') \leq d(p, p_*) \leq d(p, p'). \quad (4)$$

**Remark 3.** Not only this approximation is useful for problem (P), but the case of computing the distance between two convex hulls, the polytope distance problem. As is well known the Minkowski difference of two convex hulls is a polytope whose shortest vector has norm equal to the distance between the two polytopes, see e.g. Clarkson [10] and Gärtner and Jaggi [15].

## 1.2 The Geometry of the Triangle Algorithm

To arrive at the distance duality mentioned above, we first prove a characterization theorem that leads to this new theorem of the alternative for problem (P). We remark here that the distance duality theorem is distinct from the classical Farkas lemma, or Gordan theorem. To arrive at this theorem we first prove:

$p \in \text{conv}(S)$  if and only if given any point  $p' \in \text{conv}(S) \setminus \{p\}$ , there exists  $v_j$  such that  $d(p', v_j) > d(p, v_j)$ .

Next, we show the strict inequality can be replaced with  $d(p', v_j) \geq d(p, v_j)$ . Thus the contrapositive theorem is:

$p \notin \text{conv}(S)$  if and only if there exists  $p' \in \text{conv}(S)$  such that  $d(p', v_i) < d(p, v_i)$ , for all  $i = 1, \dots, n$ .

These two results together imply the distance duality. A corollary of our characterization theorem reveals a geometric property of a set of balls. To describe this property, denote an open ball  $B$ , its closure  $\overline{B}$ , and its boundary by  $\partial B$ , i.e.

$$B = \{x \in \mathbb{R}^m : d(x, v) < r\}, \quad \overline{B} = \{x \in \mathbb{R}^m : d(x, v) \leq r\}, \quad \partial B = \{x \in \mathbb{R}^m : d(x, v) = r\}.$$

Consider a set of open balls  $B_i = \{x \in \mathbb{R}^m : d(x, v_i) < r_i\}$ ,  $i = 1, \dots, n$ , and let  $S = \{v_1, \dots, v_n\}$ .

### Intersecting Balls Property:

If  $p \in \cap_{i=1}^n \partial B_i$ , then

$$p \in \text{conv}(S) \iff (\cap_{i=1}^n B_i) \cap \text{conv}(S) = \emptyset \iff (\cap_{i=1}^n \overline{B}_i) \cap \text{conv}(S) = \{p\}.$$

In words, suppose a set of open balls have a common boundary point  $p$ . Then  $p$  lies in the convex hull of their centers, if and only if the intersection of the open balls is empty, if and only if  $p$  is the only point in the intersection of the closure of the balls. A depiction of this property for a triangle is given in Figure 2. This property suggests we can define a geometric *dual* for problem (P):

### Problem (Q) (Intersecting Balls Problem):

Suppose there exists  $p \in \cap_{i=1}^n \partial B_i$ . Determine if  $(\cap_{i=1}^n B_i) \cap \text{conv}(S)$  is nonempty.

In fact the intersecting balls problem can be stated in more generality:

### Problem (Q') (General Intersecting Balls Problem):

Suppose there exists  $p \in \cap_{i=1}^n \partial B_i$ . Determine if  $(\cap_{i=1}^n B_i)$  is nonempty.

The Triangle Algorithm results in a fully polynomial-time approximation scheme for solving problems (Q) and (Q') with the same time complexity as that for problem (P).

When a point  $p$  lies in  $\text{conv}(S) \cap (\cap_{i=1}^n \partial B_i)$ , the union of the balls,  $\cup_{i=1}^n B_i$  is referred as the *forbidden zone* of the convex hull of the centers, see [11] or [12] for the definition and some of its properties. The notion of forbidden zone of a convex set is significant and intrinsic in the characterization of the so-called *mollified zone diagrams*, a variation of *zone diagram* of a finite set of points in the Euclidean plane, see [12]. The notion of zone diagram, introduced by Asano et al [1], is itself a very rich and interesting variation of the classical Voronoi diagram, see e.g. [2], [34]. Forbidden zones help give a characterization of mollified zone diagrams, in particular a zone diagram, [12]. For some geometric properties of forbidden zones of polygons and polytopes, see [4].

### 1.3 Comparison of Triangle Algorithm and Sparse Greedy Approximation

Formally, the distance between  $p$  and  $\text{conv}(S)$  is defined as

$$\Delta = \min\{d(p', p) : p' \in \text{conv}(S)\} = d(p_*, p). \quad (5)$$

We have,  $p \notin \text{conv}(S)$ , if and only if  $\Delta > 0$ . While problem (P) does not require the computation of  $\Delta$  when it is positive, in some applications this distance is required. However, as stated in (4), any witness approximates  $\Delta$  to within a factor of two. This fact indicates another useful property of the Triangle Algorithm.

One of the best known algorithms for determining the distance between two convex polytopes is Gilbert's algorithm, [16]. The connections and equivalence of Gilbert's algorithm and Frank-Wolfe algorithm, a gradient descent algorithm, when applied to the minimization of a convex quadratic over a simplex is formally studied in Gärtner and Jaggi [15]. They make use of a notion called *coreset*, previously studied in [10], and define a notion of  $\epsilon$ -approximation which is different from our notion given in (1). Furthermore, from the description of Gilbert's algorithm in [15] it does not follow that Gilbert's algorithm and the Triangle Algorithm are identical. However, there are similarities in theoretical performance of the two algorithms and we will discuss these next. Indeed we believe that the simplicity of the Triangle Algorithm and the new duality theorem that inspires the algorithm, as well as the its theoretical performance makes it distinct from other algorithms for the convex hull decision problem. Furthermore, these features of the Triangle Algorithm may encourage and inspire new applications of the algorithm and further theoretical analysis, in particular amortized complexity of the Triangle Algorithm. In upcoming reports we shall present some such results.

The convex hull decision problem, and its optimization form in (5) can equivalently be formulated as the minimization of a convex quadratic function over a simplex:

$$\min \left\{ f(x) = d\left(\sum_{i=1}^n x_i v_i, p\right)^2 : x \in \Sigma \right\}, \quad \Sigma = \left\{ x \in \mathbb{R}^n : \sum_{i=1}^n x_i = 1, x_i \geq 0, i = 1, \dots, n \right\}. \quad (6)$$

A greedy algorithm for the above optimization (as well as more general smooth convex function  $f(x)$ ), as described in Clarkson [10], is the following (we have changed concave maximization to convex minimization):

#### Greedy Algorithm

- **Step I.** Given  $x' \in \Sigma$ , let  $j$  be the index satisfying  $\frac{\partial f(x')}{\partial x_j} = \min\{\frac{\partial f(x')}{\partial x_i}, i = 1, \dots, n\}$ .
- **Step II.** Compute  $x'' = \text{argmin}\{f(x' + \alpha(e_j - x')) : \alpha \in [0, 1]\}$ , where  $e_j$  is the  $j$ -th vector of the standard basis. Replace  $x'$  with  $x''$ , go to Step I.

Step 1 of the Triangle Algorithm and Step I of the Greedy Algorithm (also known as sparse greedy approximation) have in common the fact that they select an index  $j$  so that  $v_j$  will be used as a pivot point. Having computed such a pivot point, Step 2 of the Triangle algorithm and Step II of the Greedy Algorithm simply perform a line search. However, the motivation behind the selection of the index  $j$  is very different. The Greedy Algorithm coincides with Frank-Wolfe algorithm and Gilbert's algorithm. The Greedy Algorithm is algebraically motivated, while the Triangle Algorithm is geometrically motivated. The Triangle

Algorithm does not need to search over all the indices to find a pivot point  $v_j$ . In its best case it finds such  $j$  in one iteration over the indices by performing only  $O(m)$  arithmetic operations. In the worst-case an iteration will require  $O(mn)$  operations. The Greedy Algorithm in contrast would require  $O(mn)$  arithmetic operations in every iteration. This can be seen when  $f(x)$  is written as  $d(Ax, p)^2$ . Its gradient at a point would in particular require the computation of  $A^T Ax$ .

The Greedy Algorithm generates a sequence of vectors  $x_{(k)} \in \mathbb{R}^n$  where  $x_{(k+1)}$  has at most  $k$  nonzero coordinates. This is advantageous when  $n$  is very large. As will be easily verifiable this property also holds for the Triangle Algorithm when the initial iterate  $p_0$  is taken to be sparse. Another property of the Greedy Algorithm is that if  $x_*$  is the optimal solution of (6), then

$$f(x_{(k)}) - f(x_*) \leq \frac{C_f}{k} = O\left(\frac{1}{k}\right), \quad (7)$$

where  $C_f$  is a constant that depends on the Hessian of  $f$ , see Clarkson [10] and Zhang [39].

The Triangle Algorithm generates a sequence of points  $p'_{(k)}$  in  $\text{conv}(S)$  that get closer and closer to  $p$ . This sequence corresponds to a sequence  $x'_{(k)}$  in  $\Sigma$ , where  $f(x'_{(k)}) = d(p'_{(k)}, p)^2$ . If  $p \in \text{conv}(S)$ ,  $f(x_*) = 0$ . Given any  $\epsilon \in (0, 1)$ , the Triangle Algorithm in  $K_\epsilon \leq 48\epsilon^{-2}$  iterations will generate a point  $p_\epsilon \in \text{conv}(S)$  satisfying

$$d(p'_{(K_\epsilon)}, p) \leq \epsilon R, \quad R = \max \{d(p, v_i), i = 1, \dots, n\}. \quad (8)$$

Given an index  $k$ , by reversing the the role of  $k$  and  $\epsilon$  and solving for  $\epsilon$  in the equation  $48/\epsilon^2 = k$ , we get  $\epsilon = \sqrt{48/k}$  so that we may write

$$d(p'_{(k)}, p) \leq \sqrt{\frac{48}{k}} R. \quad (9)$$

Equivalently,

$$f(x'_{(k)}) \leq \frac{48R^2}{k} = O\left(\frac{1}{k}\right). \quad (10)$$

In summary, when  $p \in \text{conv}(S)$  the Triangle Algorithm works similar to the Greedy Algorithm but it may perform faster because it only needs to find a pivot point  $v_j$  as opposed to finding the minimum of partial derivatives in Step I of the Greedy Algorithm. When  $p$  is not in  $\text{conv}(S)$ , the Triangle Algorithm can use any witness to give an approximation of closest point to within a factor of two, see (4). We may conclude that the Triangle Algorithm in theory is at least as effective as the Greedy Algorithm, and possibly faster whether approximating  $p \in \text{conv}(S)$ , or estimating the distance  $\Delta$  to within a factor of two.

In the context support vector machines (SVM) (see [3] for applications), Har-Peled et al. [18] use coresets to give an approximation algorithm, see also Zimak [40]. We mention these because we feel that the Triangle Algorithm, despite some similarities with existing algorithms or their analysis, is distinct from them. It is quite simple and geometrically inspired by the distance duality, a simple but new and rather surprising property.

## 1.4 Solving LP Feasibility by the Triangle Algorithm

The LP feasibility problem is to test if the polyhedron

$$\Omega = \{x \in \mathbb{R}^n : Ax = b, \quad x \geq 0\} \quad (11)$$

is nonempty, where  $A$  is an  $m \times n$  real matrix. The set of *recession directions* of  $\Omega$  is the set

$$\text{Res}(\Omega) = \{d : Ad = 0, \quad d \geq 0, \quad d \neq 0\}. \quad (12)$$

If  $A = [a_1, \dots, a_n]$ , it is trivial to show

$$\text{Res}(\Omega) = \emptyset \iff 0 \notin \text{conv}(\{a_1, \dots, a_n\}). \quad (13)$$

When  $\text{Res}(\Omega) = \emptyset$ , it is easy to prove  $\Omega \neq \emptyset$  if and only if  $0 \in \text{conv}(\{a_1, \dots, a_n, -b\})$ . In this case, given any  $\epsilon \in (0, 1)$  we can apply the Triangle Algorithm to get an  $\epsilon$ -approximate solution:

$$p' = \sum_{i=1}^n \alpha_i a_i - \alpha_{n+1} b \in \text{conv}(\{a_1, \dots, a_n, -b\}) \quad (14)$$

where

$$d(p, 0) \leq \epsilon R', \quad (15)$$

with

$$R' = \max\{d(a_1, 0), \dots, d(a_n, 0), d(b, 0)\}. \quad (16)$$

Setting  $x_0 = (\alpha_1/\alpha_{n+1}, \dots, \alpha_n/\alpha_{n+1})^T$ ,  $x_0 \geq 0$  and from (15) we have

$$d(Ax_0, b) \leq \frac{\epsilon R'}{\alpha_{n+1}}. \quad (17)$$

However, a lower bound on  $\alpha_{n+1}$  is needed in order to estimate the quality of  $x_0$  as an approximate solution of  $\Omega$ . A theoretical lower bound is the quantity:

$$\Delta_0 = \min\{d(p, 0) : p \in \text{conv}(\{a_1, \dots, a_n\})\} = \min\{d(Ax, 0) : \sum_{i=1}^n x_i = 1, x \geq 0\}, \quad (18)$$

A computational lower bound to  $\Delta_0$  can be derived by applying the Triangle Algorithm itself. From the property stated in (4), any witness  $p' \in \text{conv}(\{a_1, \dots, a_n\})$ , i.e. satisfying  $d(p', a_i) < d(0, a_i)$ , for all  $i = 1, \dots, n$ , implies

$$\frac{1}{2}d(p', 0) \leq \Delta_0 \leq d(p', 0). \quad (19)$$

We prove a sensitivity theorem that given any lower bound  $\Delta'_0$  to  $\Delta_0$ , gives the accuracy to which we need to solve the convex hull problem corresponding to testing if  $0 \in \text{conv}(\{a_1, \dots, a_n, -b\})$ . Setting  $b_0 = d(b, 0)$ , our sensitivity theorem implies that for any  $\epsilon \leq \Delta'_0/2R'$ , we have

$$d(Ax_0, b) \leq \epsilon' R', \quad \epsilon' = 2 \left(1 + \frac{b_0}{\Delta'_0}\right) \quad (20)$$

From (20) it follows that if for a given  $\epsilon_0 \in (0, 1)$  we wish to compute  $x_0 \geq 0$  such that  $d(Ax_0, b) \leq \epsilon_0 R'$ , it suffices to have  $\epsilon$  satisfying

$$\epsilon \leq \frac{\Delta'_0}{2} \min \left\{ \frac{1}{R'}, \frac{\epsilon_0}{\Delta'_0 + b_0} \right\}. \quad (21)$$

In particular, it suffices to pick  $\epsilon \leq \epsilon_0 \Delta'_0/4R'$ . Applying these, we offer a two-phase Triangle Algorithm for solving the feasibility problem in LP with the assumption that  $0 \notin \text{conv}(\{a_1, \dots, a_n\})$ :

**Two-Phase Triangle Algorithm** ( $A = [a_1, \dots, a_n]$ ,  $b$ )

- **Phase 1.** Call **Triangle Algorithm**( $\{a_1, \dots, a_n\}, 0$ ) to get a witness  $p' \in \text{conv}(\{a_1, \dots, a_n\})$ .
- **Phase 2.** Starting with  $p'$  in Phase 1, call **Triangle Algorithm**( $\{a_1, \dots, a_n, -b\}, 0$ ).

In particular, the number of iterations in Phase 2 of the algorithm is  $O(\epsilon^{-2}) = O(\epsilon_0^{-2}(R'/\Delta'_0)^2)$ .

**Remark 4.** Another lower bound to  $\Delta_0$  can be argued to be of the order of  $2^{-O(L)}$ , where  $L$  is the size of encoding of  $A$  and  $b$ . For such bound coming from LP-type analysis, see [23].

## 1.5 Problem (P) and Linear Programming Algorithms

Linear programming has found numerous practical and theoretical applications in applied mathematics, computer science, operation research and more. In particular, the simplex method of Dantiz is not only a significant algorithm for solving LP but also a theoretical tool to prove many results. Ever since the Klee-Minty [29] example showed exponential worst-case time complexity of the simplex method, many LP algorithms have been invented. The trend will most likely continue.

Problem (P) is a very special case of the LP feasibility problem. However, in fact the general LP with integer inputs can be formulated as a homogeneous case of problem (P), i.e.  $p = 0$ . The corresponding problem (P), may be referred as *homogeneous feasibility problem* (HFP), see [22]. A classical duality corresponding to HFP is Gordan's theorem, a special case of the separating hyperplane theorem, easily provable from Farkas lemma: either  $0 \in \text{conv}(S)$ , or there exists  $y \in \mathbb{R}^m$  such that  $y^T v_i > 0$  for all  $i = 1, \dots, n$ , see e.g. Chvátal [7]. It can be justified that both Khachiyan and Karmarkar algorithms explicitly or implicitly are designed to solve HFP. This is because on the one hand Karmarkar's canonical formulation of LP can easily be converted into an HFP. On the other hand, Khachiyan's ellipsoid algorithm solves a system of strict inequalities ( $Ax < b$ ) whose alternative system, by Gordan's theorem is an HFP ( $A^T y = 0, b^T y + s = 0, \sum y_i + s = 1, y \geq 0, s \geq 0$ ). For this and additional results on the connections between HFP and LP feasibility, see [19].

By exploring the close relationship between HFP which is equivalent to finding a nontrivial nonnegative zero of a quadratic form, and the diagonal matrix scaling problem, Khachiyan and Kalantari [28] have given a very simple path-following polynomial time algorithm for LP as well as for quasi doubly stochastic diagonal scaling of a positive semidefinite matrix. In particular, the algorithm can test the existence of an  $\epsilon$ -approximate solution of HFP in a number of arithmetic operations proportional to  $n^{3.5}$  and  $\ln \epsilon^{-1}$ . As approximation schemes, all known polynomial-time algorithms for LP have a complexity that is polynomial in the dimension of the data and in  $\ln \epsilon^{-1}$ . As is well known, an exact solution for an LP with integral input can be computed by rounding any approximate solution having sufficient precision. Even if the complexity of a polynomial-time algorithms for LP would allow solving problem (P) to within  $\epsilon$  accuracy in  $O(m^2 n \ln \epsilon^{-1})$  arithmetic operations, the Triangle Algorithm still offers an attractive alternative when the dimensions of the problems are large.

Other algorithms for LP include, Megiddo's algorithm [32] with a running time that for fixed  $m$  is linear in  $n$ , however, has exponential complexity in  $m$ . Since Megiddo's work a number of randomized LP algorithms have been devised, e.g. Dyer and Frieze [13], Clarkson [8], Seidel [36], Sharir and Welzel [37]. Kalai [20] gave a randomized LP simplex method with subexponential complexity bound. Matoušek, Sharir, and Welzl [30] proved another randomized subexponential complexity algorithm for LP. See also Motawani and Raghavan [33]. Kelner and Spielman [26] have given the first randomized polynomial-time simplex method that analogous to the other known polynomial-time algorithms for linear programming has a running time dependent polynomially on the bit-length of the input. A history of linear programming algorithms from computational, geometric, and complexity points of view that includes simplex, ellipsoid, and interior-point methods is given in Todd [38].

## 1.6 Outline of Results

The remaining sections of the article are as follows. In Section 2, we prove several characterization theorems that lead to a new duality, the distance duality. We then describe the associated geometric properties and problems. In Section 3, using purely geometric arguments, we analyze the worst-case reduction of the gap in going from a new approximation to the next. Using the results in Section 3, in Section 4 we describe the Triangle Algorithm and analyze its worst-case complexity. In Section 5, we apply the Triangle Algorithm to derive a complexity for solving an LP feasibility with the assumption of that the problem, if feasible, has no recession direction. In Section 6, we give the analysis of the Triangle Algorithm for solving an LP feasibility where it is not known whether or not it has a recession direction. This corresponds to solving a general LP problem. The purpose and significance of the results in Section 6 lies in the simplicity of the Triangle Algorithm, whether or not in practice we would actually want to solve a general LP with the suggested approach.



## 2 Characterizations and Applications

Throughout the section, let  $S = \{v_1, \dots, v_n\}$  be a set of points in  $\mathbb{R}^m$ , and let  $p \in \mathbb{R}^m$ .

**Theorem 1. (Characterization of Feasibility)**  $p \in \text{conv}(S)$  if and only if given any  $p' \in \text{conv}(S) \setminus \{p\}$ , there exists  $v_j \in S$  such that  $d(p', v_j) > d(p, v_j)$ .

*Proof.* Suppose  $p \in \text{conv}(S)$ . Consider the Voronoi cell of  $p$  with respect to the two point set  $\{p, p'\}$ , i.e.  $V(p) = \{x \in \mathbb{R}^m : d(x, p) < d(x, p')\}$  (see Figure 1). We claim there exists  $v_j \in V(p)$ . If not,  $S$  is a subset of  $\overline{V}(p') = \{x \in \mathbb{R}^m : d(x, p) \leq d(x, p')\}$ . But since  $\overline{V}(p')$  is convex it contains  $\text{conv}(S)$ . But since  $V(p) \cap \overline{V}(p') = \emptyset$  this contradicts that  $p \in \text{conv}(S)$ .

Conversely, suppose that for any  $p' \in \text{conv}(S) \setminus \{p\}$  there exists  $v_j$  such that  $d(p', v_j) > d(p, v_j)$ . If  $p \notin \text{conv}(S)$ , let  $p' \in \text{conv}(S)$  be the closest point to  $p$ . Since the closest point is unique, for each  $i = 1, \dots, n$ , in the triangle  $\triangle pp'v_i$  the angle  $\angle pp'v_i$  must be non-acute. Hence,  $d(p', v_i) < d(p, v_i)$  for all  $i$ , a contradiction. Thus,  $p \in \text{conv}(S)$ .  $\square$

The following is a convenient restatement of Theorem 1, relaxing the strict inequality,  $d(p', v_j) > d(p, v_j)$ . It has an identical proof to that theorem.

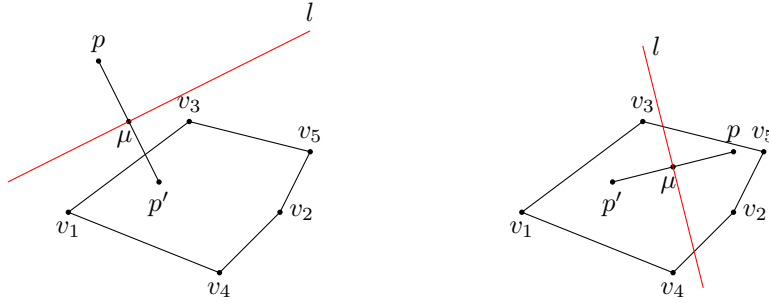


Figure 1: Example of cases where orthogonal bisector of  $pp'$  does and does not separate  $p$  from  $\text{conv}(S)$ .

**Theorem 2.**  $p \in \text{conv}(S)$  if and only if given any  $p' \in \text{conv}(S) \setminus \{p\}$ , there exists  $v_j \in S$  such that  $d(p', v_j) \geq d(p, v_j)$ .  $\square$

**Definition 1.** We call a point  $p' \in \text{conv}(S)$  a *witness* if  $d(p', v_i) < d(p, v_i)$ , for all  $i = 1, \dots, n$ . We denote the set of all witnesses by  $W_p$ .

The following is a characterization of infeasibility.

**Theorem 3. (Characterization of Infeasibility)**  $p \notin \text{conv}(S)$  if and only if there exists a witness  $p'$ .

*Proof.* Suppose  $p \notin \text{conv}(S)$ . Then by Theorem 2 there exists  $p' \in \text{conv}(S)$  such that  $d(p', v_i) < d(p, v_i)$ , for all  $i = 1, \dots, n$ . But then  $p'$  is a witness. Conversely, given that  $p'$  is a witness, Theorem 1 implies  $p \notin \text{conv}(S)$ .  $\square$

Thus we may conclude the following, a new duality for the convex hull decision problem.

**Theorem 4. (Distance Duality)** Precisely one of the two conditions is satisfied:

- (i) For each  $p' \in \text{conv}(S)$  there exists  $v_j \in S$  such that  $d(p', v_j) \geq d(p, v_j)$ .
- (ii) There exists a witness.  $\square$

The following is a straightforward but geometrically appealing characterization of the set of witnesses as the intersection of open balls and  $\text{conv}(S)$ .

**Proposition 1.** Let  $B_i = \{x \in \mathbb{R}^m : d(x, v_i) < d(p, v_i)\}$ ,  $i = 1, \dots, n$ . Then  $W_p = \text{conv}(S) \cap (\cap_{i=1}^n B_i)$ . In particular,  $W_p$  is convex and lies in the relative interior of  $\text{conv}(S)$ .  $\square$

Figure 2 gives a case when  $W_p$  is empty. Figure 3 gives several scenarios when  $W_p$  is nonempty. Next, we state a corollary of Theorem 1 and Theorem 2.

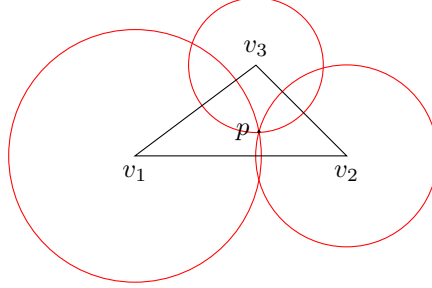


Figure 2: A case with no witnesses:  $p \in \text{conv}(S)$ .

**Corollary 1. (Intersecting Balls Property)** Consider the set of open balls  $B_i = \{x \in \mathbb{R}^m : d(x, v_i) < r_i\}$ . Assume they have a common boundary point  $p$ , i.e.  $p \in \cap_{i=1}^n \partial B_i = \{x \in \mathbb{R}^m : d(x, v_i) = r_i\}$ . Let  $S = \{v_1, \dots, v_n\}$ . Then  $p \in \text{conv}(S)$  if and only if  $(\cap_{i=1}^n B_i) \cap \text{conv}(S) = \emptyset$  (see Figure 2).

*Proof.* Suppose  $p \in \text{conv}(S)$ . Pick any point  $p' \in \text{conv}(S) \setminus \{p\}$ . Then by Theorem 1 there exists  $v_j$  such that  $d(p', v_j) > d(p, v_j)$ . But this implies  $p' \notin B_j$ , hence  $(\cap_{i=1}^n B_i) \cap \text{conv}(S) = \emptyset$ .

Conversely, suppose that  $(\cap_{i=1}^n B_i) \cap \text{conv}(S) = \emptyset$ . Thus for each  $p' \in \text{conv}(S)$  there exists  $v_j$  such that  $d(p', v_j) \geq d(p, v_j)$ . Then by Theorem 2 we have  $p \in \text{conv}(S)$ .  $\square$

**Proposition 2. (The Orthogonal Bisector Property)** Suppose  $p'$  is a witness, i.e.  $p' \in \text{conv}(S)$  satisfies  $d(p', v_i) < d(p, v_i)$ , for all  $i = 1, \dots, n$ . Then, the orthogonal bisector hyperplane of the line segment  $pp'$  separates  $p$  from  $\text{conv}(S)$ . More specifically, let  $c = p - p'$  and  $\gamma = \frac{1}{2}(p^T p - p'^T p')$ . If  $H = \{x \in \mathbb{R}^m : c^T x = \gamma\}$ , then  $p \in H_+ = \{x \in \mathbb{R}^m : c^T x > \gamma\}$  and  $\text{conv}(S) \subset H_- = \{x \in \mathbb{R}^m : c^T x < \gamma\}$ .

*Proof.* It is easy to verify that  $p \in H_+$ . We claim  $v_i \in H_-$ , for each  $i$ . For each  $i = 1, \dots, n$  we have,  $d^2(p', v_i) < d^2(p, v_i)$ . Equivalently,  $(p' - v_i)^T (p' - v_i) < (p - v_i)^T (p - v_i)$ . Simplifying, gives

$$2(p - p')^T v_i < (p^T p - p'^T p').$$

Hence  $S \subset H_-$ . Since  $H_-$  is convex, any convex combination of points in  $S$  is also in  $H_-$ .  $\square$

We now give a complete characterization of the witness set.

**Theorem 5. (Characterization of Witness Set)**  $p' \in W_p$  if and only if the orthogonal bisector hyperplane of the line segment  $pp'$  separates  $p$  from  $\text{conv}(S)$ .

*Proof.* By Proposition 2,  $p' \in W_p$  implies the orthogonal bisector hyperplane of the line segment  $pp'$  separates  $p$  from  $\text{conv}(S)$ . Conversely, suppose for some  $p' \in \text{conv}(S)$  the orthogonal bisector hyperplane of the line segment connecting  $pp'$  separates  $p$  from  $\text{conv}(S)$ . Then, in particular we have  $d(p', v_i) < d(p, v_i)$ , for all  $i = 1, \dots, n$ . Hence,  $p' \in W_p$ .  $\square$

**Corollary 2.** Suppose  $p \notin \text{conv}(S)$ . Let

$$\Delta = d(p, \text{conv}(S)) = \min\{d(p, x) : x \in \text{conv}(S)\}.$$

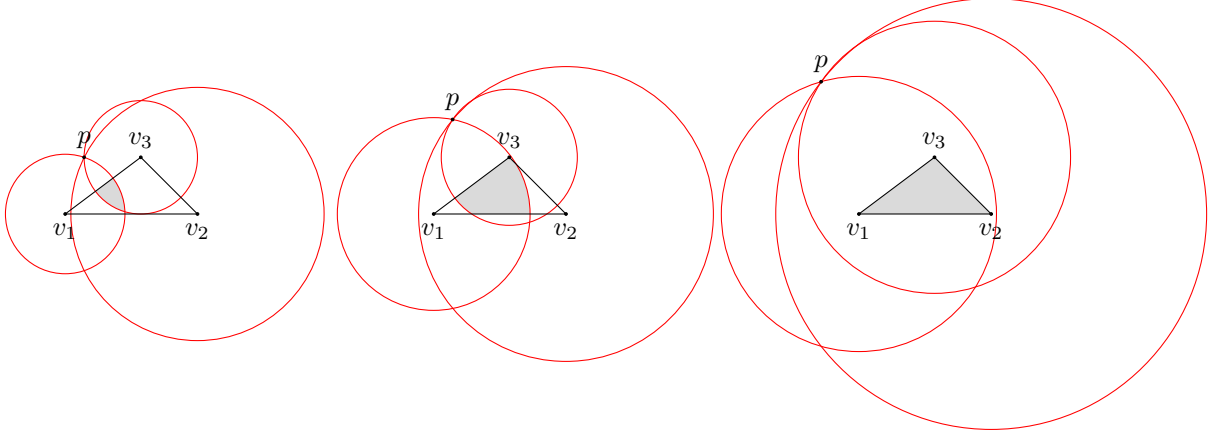


Figure 3: Examples of nonempty witness set  $W_p$ , gray areas:  $p \notin \text{conv}(S)$ .

Then any  $p' \in W_p$  gives an estimate of  $\Delta$  to within a factor of two. More precisely,

$$\frac{1}{2}d(p, p') \leq \Delta \leq d(p, p').$$

*Proof.* The inequality  $\Delta \leq d(p, p')$  is obvious. The first inequality follows from Theorem 5.  $\square$

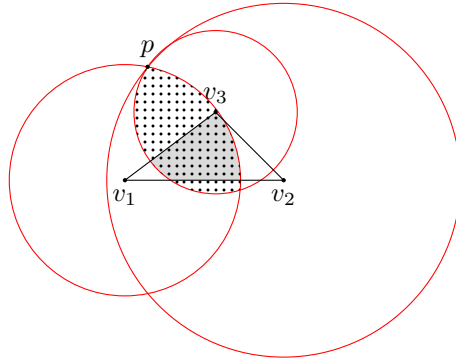


Figure 4: The set  $\overline{W}_p$  of general witness (dotted area) is a superset of  $W_p$  (gray area).

Before we utilize the characterization theorems proved here we wish to give a variation of Theorem 1. The theorem shows that the notion of a witness need not be restricted to the convex hull of  $S$ . The proof is identical to the proof of Theorem 1 and is omitted.

**Theorem 6.**  $p \in \text{conv}(S)$  if and only if given any point  $p' \neq p$ , there exists  $v_j \in S$  such that  $d(p', v_j) > d(p, v_j)$ .  $\square$

We may thus give a more general distance duality as well as definition for a witness.

**Definition 2.** We call a point  $p' \in \mathbb{R}^m$  a *general witness* if  $d(p', v_i) < d(p, v_i)$ , for all  $i = 1, \dots, n$ . We denote the set of all general witnesses by  $\overline{W}_p$ .

Figure 4 depicts the set  $\overline{W}_p$  which of course contains  $W_p$  as a subset. The following is a corollary of Theorem 2 and Theorem 6

**Corollary 3. (General Intersecting Balls Property)** *Consider the set of open balls  $B_i = \{x \in \mathbb{R}^m : d(x, v_i) < r_i\}$ . Assume  $p \in \cap_{i=1}^n \partial B_i$ . Then  $p \in \text{conv}(\{v_1, \dots, v_n\})$  if and only if  $(\cap_{i=1}^n B_i) = \emptyset$ .*

*Proof.* Suppose  $p \in \text{conv}(S)$ . Pick any point  $p' \neq p$ . Then by Theorem 6 there exists  $v_j$  such that  $d(p', v_j) > d(p, v_j)$ . But this implies  $p' \notin B_i$ , hence  $(\cap_{i=1}^n B_i) = \emptyset$ .

Conversely, suppose that  $(\cap_{i=1}^n B_i) = \emptyset$ . In particular, for each  $p' \in \text{conv}(S)$  there exists  $v_j$  such that  $d(p', v_j) \geq d(p, v_j)$ . Then by Theorem 2 we have  $p \in \text{conv}(S)$ .  $\square$

**Remark 5.** The general open balls property suggests that in proving the infeasibility of  $p$  we have the freedom of choosing a witnesses outside of the convex hulls of  $S$ . However, algorithmically it may have no advantage over the Triangle Algorithm to be formally described later.

### 3 Reduction of Gap and Its Worst-Case Analysis

In what follows we state a theorem that is fundamental in the analysis of the algorithm to be described in the next section. It relates to one iteration of the algorithm to test if  $p$  lies in  $\text{conv}(S)$  and reveals its worst-case performance. In the theorem below the reader may consider  $p'$  as a given point in  $\text{conv}(S)$  and  $v$  as a point in  $S$  to be used as a *pivot point* in order to compute a new point  $p''$  in  $\text{conv}(S)$  where the new gap  $d(p'', p)$  is to be a reduction of the current gap  $d(p', p)$ .

**Theorem 7.** *Let  $p, p', v$  be distinct points in  $\mathbb{R}^m$ . Suppose  $d(p, v) \leq d(p', v)$ . Let  $p''$  be the point on the line segment  $p'v$  that is closest to  $p$ . Assume  $p'' \neq v$ . Let  $\delta = d(p', p)$ ,  $\delta' = d(p'', p)$ , and  $r = d(p, v)$  (see Figure 5). Then,*

$$\delta' \leq \begin{cases} \delta \sqrt{1 - \frac{\delta^2}{4r^2}}, & \text{if } \delta \leq r; \\ r, & \text{otherwise.} \end{cases} \quad (22)$$

*Proof.* Given  $\delta \leq r$ , consider  $p'$  as a variable  $x'$  and the corresponding  $p''$  as  $x''$ . We will consider the maximum value of  $d(x'', p)$  subject to the desired constraints. We will prove

$$\delta^* = \max \{d(x'', p) : x \in \mathbb{R}^m, \quad d(x', p) = \delta, \quad d(x', v) \geq r\} = \delta \sqrt{1 - \frac{\delta^2}{4r^2}}. \quad (23)$$

This optimization problem can be stated in the two-dimensional Euclidean plane. Assume  $p \neq p''$ , and consider the two-dimensional plane that passes through the points  $p, p', v$ . Given that  $\delta \leq r$ ,  $p'$  must lie inside or on the boundary of the circle of radius  $\delta$  centered at  $p$ , but outside or on the boundary of the circle of radius  $r$  centered at  $v$ , see Figure 5, circles  $C$ ,  $C'$ , and  $C''$ .

Now consider the circle of radius  $\delta$  centered at  $p$ ,  $C''$  in Figure 5. Consider the ratio  $\delta'/r$  as  $p'$  ranges over all the points on the circumference of  $C''$  while outside or on the boundary of  $C'$ . It is geometrically obvious and easy to argue that this ratio is maximized when  $p'$  is a point of intersection of the circles  $C'$  and  $C''$ , denoted by  $x^*$  in Figure 6. We now compute the corresponding ratio.

Considering Figure 6, and the isosceles triangle  $\triangle vpx^*$ , let  $h$  denote the length of the bisector line from  $v$  to the base, and let  $q$  denote the midpoint of  $p$  and  $x^*$ . Consider the right triangles  $\triangle pvq$  and  $\triangle px^*x^{**}$ . The angle  $\angle vpq$  is identical with  $\angle px^*x^{**}$ . Hence, the two triangles are similar and we may write

$$\frac{\delta^*}{\delta} = \frac{h}{r} = \frac{1}{r} \sqrt{r^2 - \frac{\delta^2}{4}} = \sqrt{1 - \frac{\delta^2}{4r^2}}. \quad (24)$$

This proves the first inequality in (22).

Next, suppose  $\delta > r$ . Figure 7 considers several possible scenarios for this case. If in the triangle  $\triangle pvp'$  the angle  $\angle pvp'$  is acute, the line segment  $p'v$  must necessarily intersect  $C'$ . This implies  $p''$  is the bisector of a chord in  $C'$ , hence inside of  $C'$ . If the angle  $\angle pvp'$  is at least  $\pi/2$ , then  $p''$  will coincide with  $v$ . Hence, proving the second inequality in (22).  $\square$

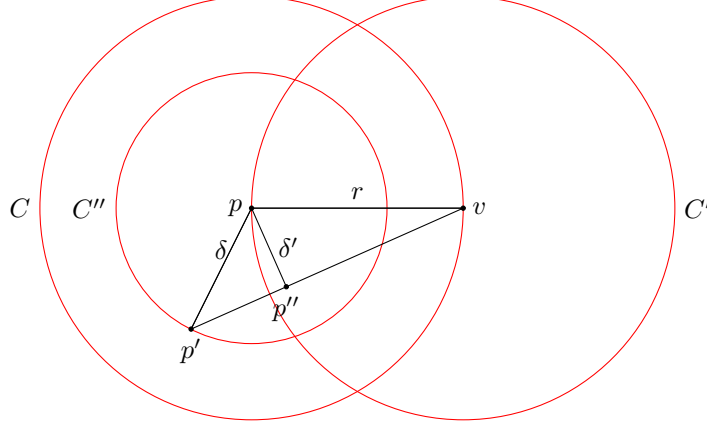


Figure 5: Depiction of gaps  $\delta = d(p', p)$ ,  $\delta' = d(p'', p)$ , when  $\delta \leq r = d(p, v)$ .

## 4 The Triangle Algorithm and Its Analysis

In this section we describe a simple algorithm for solving problem (P). For convenience we shall refer to this algorithm as the *Triangle Algorithm*. The justification in the name lies in the fact that in each iteration the algorithm searches for a triangle  $\triangle pp'v_j$  where  $v_j \in S$ ,  $p' \in \text{conv}(S) \setminus \{p\}$ , such that  $d(p', v_j) \geq d(p, v_j)$ . Given that such triangle exists, it uses  $v_j$  as a pivot point to “pull” the current iterate  $p'$  closer to  $p$  to get a new iterate  $p'' \in \text{conv}(S)$ . If no such a triangle exists, then by Theorem 3,  $p'$  is a witness certifying that  $p$  is not in  $\text{conv}(S)$ . The Triangle Algorithm consists of iterating the following two steps:

**Triangle Algorithm** ( $S = \{v_1, \dots, v_n\}, p$ )

- **Step 1.** Given  $p' \in \text{conv}(S) \setminus \{p\}$ , check if there exists a *pivot point*  $v_j \in S$ , i.e.  $d(p', v_j) \geq d(p, v_j)$ . If no such  $v_j$  exists, then  $p'$  is a *witness*, stop.
- **Step 2.** Otherwise, compute the *step-size*

$$\alpha = \frac{(p - p')^T (v_j - p')}{d^2(v_j, p')}. \quad (25)$$

Let the *iterate* be defined as

$$p'' = \begin{cases} (1 - \alpha)p' + \alpha v_j, & \text{if } \alpha \in [0, 1]; \\ v_j, & \text{otherwise.} \end{cases} \quad (26)$$

Replace  $p'$  with  $p''$ , go to Step 1.

By an easy calculation that shift  $p'$  to the origin, it follows that the point  $p''$  in Step 2 is the closest point to  $p$  on the line  $p'v_j$ . Since  $p''$  is a convex combination of  $p'$  and  $v_j$  it will remain in  $\text{conv}(S)$ . The algorithm replaces  $p'$  with  $p''$  and repeats the above iterative step. Note that a pivot point  $v_j$  may or may not be a vertex of  $\text{conv}(S)$ . In the following we state some basic properties of the algorithm to be used in the analysis of its complexity.

**Proposition 3.** *The algorithm satisfies the following properties:*

- (i) *In each iteration of Step 2 if  $p'' \neq v_j$ , the step-size  $\alpha$  lies in  $(0, 1)$ .*

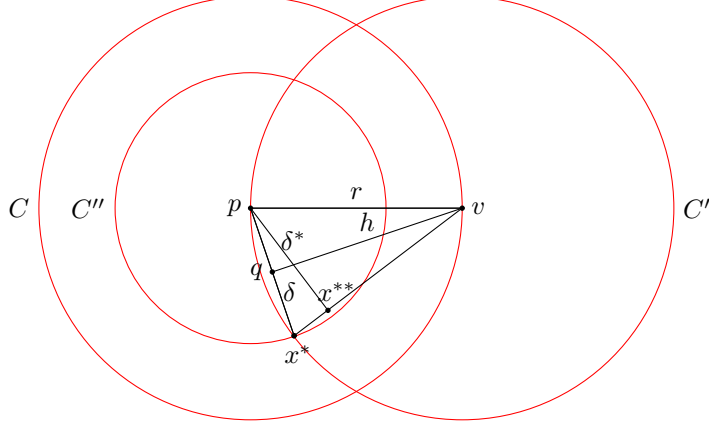


Figure 6: The worst-case senario for the gap  $\delta' = \delta^*$ , when  $\delta \leq r$ .

(ii) Given an explicit representation of  $p'$  as a convex combination of  $v_i$ 's,

$$p' = \sum_{i=1}^n \alpha_i v_i, \quad \sum_{i=1}^n \alpha_i = 1, \quad \alpha_i \geq 0, \quad \forall i, \quad (27)$$

$p''$  can also be explicitly written as a convex combination of  $v_i$ 's,

$$p'' = \sum_{i=1}^n \beta_i v_i, \quad \beta_j = (1 - \alpha)\alpha_j + \alpha, \quad \beta_i = (1 - \alpha)\alpha_i, \quad \forall i \neq j. \quad (28)$$

(iii) Each iteration of the algorithm take at most  $O(mn)$  arithmetic operations and comparisons.

(iv) Each  $v_i$  can be selected as an iterate  $p''$  at most once.

(v) If  $v_j$  is selected as a pivot point more than once, then except possibly for its first selection as an iterate, in any subsequent selection of  $v_j$  as a pivot point the iterate  $p_k$  will satisfy  $d(p, p_k) \leq d(p, v_j)$ .

*Proof.* To prove (i), note that we may have  $p'' = v_j$ , occurring when the line  $pv$  and  $p'v$  are orthogonal or when they make an obtuse angle (see Figure 7, rightmost case of  $p'$ ). Since  $p' \neq p$ ,  $p''$  cannot coincide with  $p'$ . Otherwise, this would imply in the triangle  $\triangle pp'v$  the angle  $\angle pp'v$  is non-acute, implying  $d(p, v_j) > d(p', v_j)$ . But this contradicts that  $v_j$  is a pivot point. Given that  $p'' \neq v_j$ , in the triangle  $\triangle p'pv_j$  the angle  $\angle p'pv_j$  is obtuse. Therefor, if  $p'' \neq v_j$ ,  $0 < \alpha < 1$ .

The proof of (ii) is straightforward from the fact that  $0 < \alpha \leq 1$  since this implies  $\beta_i \geq 0$  for all  $i$  and they sum up to one.

The proof of (iii) is obvious since in the worst-case in each iteration we need to compute and compare  $d(p, v_i)$  and  $d(p', v_i)$  for  $i = 1, \dots, n$ .

The proof of (iv) follows from the fact that the sequence of distances  $d(p, p_k)$  is a decreasing sequence.

To prove (v), assume  $v_j$  is selected as a pivot and  $d(p, p_k) > d(p, v_j)$ . If  $v_j$  becomes an iterate, then the proof follows from (iv). Otherwise, from Theorem 7, if  $d(p_k, v_j) > d(p, v_j)$ , then  $d(p, p_{k+1}) \leq d(p, v_j)$ . Then, the monotonicity property of the sequence of distances implies the desired result.  $\square$

We are bow ready to analyze the complexity of the algorithm. Set

$$R = \max \{d(p, v_i), i = 1, \dots, n\}. \quad (29)$$

**Corollary 4.** Let  $p, p', p'', v$  be as in Theorem 7, and  $r = d(p, v)$ ,  $\delta = d(p, p')$ ,  $\delta' = d(p, p'')$ . If  $p'' \neq v$  and  $\delta \leq r$ , then

$$\delta' \leq \delta \sqrt{1 - \frac{\delta^2}{4R^2}} \leq \delta \exp\left(-\frac{\delta^2}{8R^2}\right). \quad (30)$$

*Proof.* The first inequality follows from Theorem 7 and the definition of  $R$ . To prove the next inequality, we use that  $1 + x \leq \exp(x)$ , and set  $x = -\delta^2/4R^2$ .  $\square$

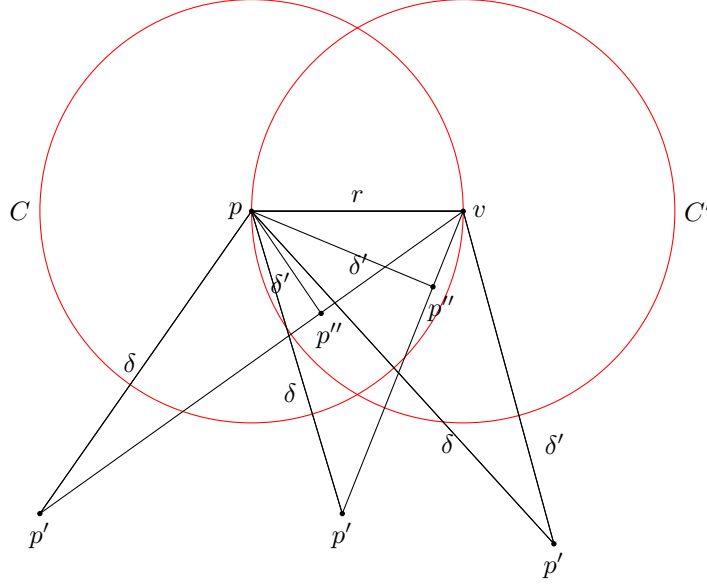


Figure 7: Depiction of gaps  $\delta = d(p', p)$ ,  $\delta' = d(p'', p)$ , when  $\delta > r = d(p, v)$ .

**Remark 6.** The analysis of complexity of the Triangle Algorithm will make repeated use of (30) in Corollary 4. By part (v) of Proposition 3, the number of iterations where an element  $v_j \in S$  is selected as an iterate is at most  $n$ . Therefore, except for  $n$  iterations, in any other iteration of the algorithm the inequality  $\delta \leq r$  is satisfied so that (30) will apply. Hence, for the sake of simplicity in the forgoing complexity analysis we will exclude the occurrence of these exceptional iterations.

**Lemma 1.** Assume  $p$  is in  $\text{conv}(S)$ . Pick  $p_0 \in \text{conv}(S) \setminus \{p\}$ , let  $\delta_0 = d(p_0, p)$ . Let  $k \equiv k(\delta_0)$  be the maximum number of iterations of the Triangle Algorithm in order to compute a point  $p_k$  in  $\text{conv}(S)$  so that if  $\delta_j = d(p_j, p)$  for  $j = 1, \dots, k$ , we have

$$\delta_k \leq \frac{\delta_0}{2} < \delta_j, \quad j = 1, \dots, k-1. \quad (31)$$

Then,  $k$  satisfies

$$k = k(\delta_0) \leq \lceil N_0 \rceil, \quad N_0 \equiv N(\delta_0) = (32 \ln 2) \frac{R^2}{\delta_0^2} < 23 \frac{R^2}{\delta_0^2} \quad (32)$$

*Proof.* For each  $j = 1, \dots, k-1$ , the repeated application of (30) in Corollary 4, the fact that for each such  $j$ ,  $\delta_j > \delta_0/2$ , and the monotonicity of the exponential function implies

$$\delta_j \leq \delta_{j-1} \exp\left(-\frac{\delta_{j-1}^2}{8R^2}\right) \leq \delta_{j-1} \exp\left(-\frac{\delta_0^2}{32R^2}\right) \leq \delta_{j-2} \exp\left(-\frac{2\delta_0^2}{32R^2}\right) \leq \dots \quad (33)$$

It follows that

$$\delta_{k-1} \leq \delta_0 \exp\left(-\frac{(k-1)\delta_0^2}{32R^2}\right). \quad (34)$$

Thus from (30) and (34) we have

$$\delta_k \leq \delta_{k-1} \exp\left(-\frac{\delta_{k-1}^2}{8R^2}\right) \leq \delta_0 \exp\left(-\frac{k\delta_0^2}{32R^2}\right). \quad (35)$$

To have  $\delta_k \leq \delta_0/2$ , it suffices to satisfy

$$\exp\left(-\frac{k\delta_0^2}{32R^2}\right) \leq \frac{1}{2}. \quad (36)$$

Solving for  $k$  in the above inequality implies the claimed bound in (32).  $\square$

**Theorem 8.** *The Triangle Algorithm correctly solves problem (P) as follows:*

(i) *Suppose  $p \in \text{conv}(S)$ . Given  $\epsilon > 0$ , the number of iterations  $K_\epsilon$  to compute a point  $p_\epsilon$  in  $\text{conv}(S)$  so that  $d(p, p_\epsilon) \leq \epsilon d(p, v_i)$ , for some  $v_i \in S$  satisfies*

$$K_\epsilon \leq \frac{48}{\epsilon^2} = O(\epsilon^{-2}). \quad (37)$$

(ii) *Suppose  $p \notin \text{conv}(S)$ . If  $\Delta$  denotes the distance from  $p$  to  $\text{conv}(S)$ , i.e.*

$$\Delta = \min \{d(x, p) : x \in \text{conv}(S)\}, \quad (38)$$

*the number of iterations  $K_\Delta$  to compute a witness, a point  $p_\Delta$  in  $\text{conv}(S)$  so that  $d(p_\Delta, v_i) < d(p, v_i)$  for all  $v_i \in S$ , satisfies*

$$K_\Delta \leq \left\lceil \frac{8R^2}{\Delta^2} \ln\left(\frac{2\delta_0}{\Delta}\right) \right\rceil. \quad (39)$$

*Proof.* From Lemma 1 and definition of  $k(\delta_0)$  (see (32)), in order to half the initial gap from  $\delta_0$  to  $\delta_0/2$ , in the worst-case the Triangle Algorithm requires  $k(\delta_0)$  iterations. Then, in order to reduce the gap from  $\delta_0/2$  to  $\delta_0/4$  it requires at most  $k(\delta_0/2)$  iterations, and so on. From (32), for each nonnegative integer  $r$  the worst-case number of iterations to reduce a gap from  $\delta_0/2^r$  to  $\delta_0/2^{r+1}$  is given by

$$k\left(\frac{\delta_0}{2^r}\right) \leq \left\lceil N\left(\frac{\delta_0}{2^r}\right) \right\rceil = \lceil 2^{2r} N_0 \rceil \leq 2^{2r} \lceil N_0 \rceil. \quad (40)$$

Therefore, if  $t$  is the smallest index such that  $\delta_0/2^t \leq \epsilon R$ , i.e.

$$2^{t-1} < \frac{\delta_0}{R\epsilon} \leq 2^t, \quad (41)$$

then the total number of iterations of the algorithm,  $K_\epsilon$ , to test if condition (i) is valid satisfies:

$$K_\epsilon \leq \lceil N_0 \rceil (1 + 2^2 + 2^4 + \dots + 2^{2(t-1)}) \leq \lceil N_0 \rceil \frac{2^{2t} - 1}{3} \leq \lceil N_0 \rceil 2 \times 2^{2(t-1)} \leq (N_0 + 1) \frac{2\delta_0^2}{R^2\epsilon^2} \quad (42)$$

From (32) we get

$$K_\epsilon \leq \left(23 \frac{R^2}{\delta_0^2} + 1\right) \frac{2\delta_0^2}{R^2\epsilon^2} = \left(23 + \frac{\delta_0^2}{R^2}\right) \frac{2}{\epsilon^2}. \quad (43)$$

Since  $p \in \text{conv}(S)$  and from the definition of  $R$  (see (29)) we have  $\delta_0 = d(p, p_0) \leq R$ , hence we get the claimed bound on  $K_\epsilon$  in (37).



Suppose  $p \notin \text{conv}(S)$ . With  $p_0 \in \text{conv}(S)$ , assume that the algorithm generates  $p_j \in \text{conv}(S) \setminus W_p$ . Since  $p \notin \text{conv}(S)$ , for  $j = 0, \dots, k$  we have

$$\Delta < \delta_j = d(p_j, p). \quad (44)$$

From the repeated application of (30) in Corollary 4 we have

$$\delta_k \leq \delta_{k-1} \exp\left(-\frac{\delta_{k-1}^2}{8R^2}\right) \leq \delta_{k-1} \exp\left(-\frac{\Delta^2}{8R^2}\right) \leq \delta_{k-2} \exp\left(-\frac{2\Delta^2}{8R^2}\right) \leq \dots \quad (45)$$

Thus

$$\delta_k \leq \delta_0 \exp\left(-\frac{k\Delta^2}{8R^2}\right). \quad (46)$$

To determine  $K_\Delta$  it suffices to solve

$$\delta_0 \exp\left(-\frac{k\Delta^2}{8R^2}\right) \leq \frac{\Delta}{2}. \quad (47)$$

Solving for  $k = K_\Delta$  gives the claimed bound in (39)  $\square$

## 5 Solving LP Feasibility Via The Triangle Algorithm

Consider the LP feasibility problem, testing if  $\Omega$  is nonempty, where

$$\Omega = \{x \in \mathbb{R}^n : Ax = b, \quad x \geq 0\}, \quad (48)$$

with  $A = [a_1, a_2, \dots, a_n]$ ,  $a_i \in \mathbb{R}^m$ .

It is well known that through linear programming duality, the general LP problem can be reduced to a single LP feasibility problem. In this section we show how the Triangle Algorithm for problem (P) can be modified to either prove that  $\Omega$  is empty, or to compute an approximate feasible point when the set of recession directions of  $\Omega$  is empty, where

$$\text{Res}(\Omega) = \{d : Ad = 0, \quad d \geq 0, \quad d \neq 0\}. \quad (49)$$

**Definition 3.** Given  $\epsilon$ , we shall say  $x_0 \in \mathbb{R}^n$  is a  $\epsilon$ -approximate solution (or feasible point) of  $\Omega$  if the point  $Ax_0$  lies in  $\text{Cone}(\{a_1, \dots, a_n\})$ , i.e.

$$Ax_0 \in \{Ax : x \geq 0\},$$

satisfying

$$d(Ax_0, b) \leq \epsilon R', \quad (50)$$

where

$$R' = \max\{d(a_1, 0), \dots, d(a_n, 0), d(b, 0)\}. \quad (51)$$

The following is easy to show.

**Proposition 4.** Suppose  $0 \notin \text{conv}(\{a_1, \dots, a_n\}) = \{Ax : \sum_{i=1}^n x_i = 1, x \geq 0\}$ . Then  $\Omega = \emptyset$  if and only if  $0 \in \text{conv}(\{a_1, \dots, a_n, -b\})$ .  $\square$

**Remark 7.** It is easy to see that  $\text{Res}(\Omega) = \emptyset$  if and only if  $0 \notin \text{conv}(\{a_1, \dots, a_n\})$ . In particular, if  $\text{Res}(\Omega) = \emptyset$ ,  $\Omega$  is a bounded set, possibly empty. Thus, in this case LP feasibility reduces to a single convex hull decision problem. The following sensitivity theorem establish the needed accuracy to which an approximate solution in  $\text{conv}(\{a_1, \dots, a_n, -b\})$  should be computed.

**Theorem 9. (Sensitivity Theorem)** Suppose  $0 \notin \text{conv}(\{a_1, \dots, a_n\})$ . Let

$$\Delta_0 = \min\{d(p, 0) : p \in \text{conv}(\{a_1, \dots, a_n\})\} = \min\{d(Ax, 0) : \sum_{i=1}^n x_i = 1, x \geq 0\}, \quad (52)$$

Let  $\Delta'_0$  be any number such that  $0 < \Delta'_0 \leq \Delta_0$ . Let  $b_0 = d(b, 0)$  and  $R'$  as in (51). Suppose  $\epsilon > 0$  satisfies

$$\epsilon \leq \frac{\Delta'_0}{2R'}. \quad (53)$$

Suppose we have computed

$$p' = (\alpha_1 a_1 + \dots + \alpha_n a_n) - \alpha_{n+1} b \in \text{conv}(\{a_1, \dots, a_n, -b\}) \quad (54)$$

satisfying

$$d(p', 0) \leq \epsilon R' \quad (55)$$

Let

$$x_0 = \left(\frac{\alpha_1}{\alpha_{n+1}}, \dots, \frac{\alpha_n}{\alpha_{n+1}}\right)^T. \quad (56)$$

Then,  $x_0 \geq 0$ , and if

$$\epsilon' = 2 \left(1 + \frac{b_0}{\Delta'_0}\right) \epsilon, \quad (57)$$

we have

$$d(Ax_0, b) \leq \epsilon' R', \quad (58)$$

i.e.  $x_0$  is an  $\epsilon'$ -approximate feasible point of  $\Omega$ .

*Proof.* Letting  $q' = p'/\alpha_{n+1}$ , from (54) and (56) we have

$$q' = \frac{p'}{\alpha_{n+1}} = Ax_0 - b. \quad (59)$$

From (55) we have,

$$d(q', 0) = d(Ax_0, b) \leq \frac{\epsilon R'}{\alpha_{n+1}}. \quad (60)$$

We wish to compute a lower bound on  $\alpha_{n+1}$ . Let

$$q = \frac{\alpha_1}{1 - \alpha_{n+1}} a_1 + \dots + \frac{\alpha_n}{1 - \alpha_{n+1}} a_n. \quad (61)$$

Note that  $q \in \text{conv}(\{a_1, \dots, a_n\})$ . Then, by definition of  $\Delta_0$  we have,

$$d(q, 0) \geq \Delta_0 \geq \Delta'_0. \quad (62)$$

Let

$$q'' = \frac{p'}{1 - \alpha_{n+1}} = q - \frac{\alpha_{n+1}}{1 - \alpha_{n+1}} b. \quad (63)$$

From (55) we also have

$$d(q'', 0) \leq \frac{\epsilon R'}{1 - \alpha_{n+1}}. \quad (64)$$

Applying the triangle inequality,  $d(u, 0) - d(v, 0) \leq d(u, v)$ , to (63) and then using the bound in (64) we have

$$d(q, 0) - \frac{\alpha_{n+1}}{1 - \alpha_{n+1}} d(b, 0) \leq d(q'', 0) \leq \frac{\epsilon R'}{1 - \alpha_{n+1}}. \quad (65)$$

From (65) and (62), and that  $d(b, 0) = b_0$  we get

$$\Delta'_0 \leq \frac{\alpha_{n+1}}{1 - \alpha_{n+1}} b_0 + \frac{\epsilon R'}{1 - \alpha_{n+1}}. \quad (66)$$

Equivalently,

$$\Delta'_0 - \epsilon R' \leq \alpha_{n+1}(\Delta'_0 + b_0). \quad (67)$$

From (67) and the assumption in (53) we get

$$\alpha_{n+1} \geq \frac{\Delta'_0 - \epsilon R'}{\Delta'_0 + b_0} \geq \frac{\Delta'_0}{2(\Delta'_0 + b_0)}. \quad (68)$$

Substituting the lower bound in (68) for  $\alpha_{n+1}$  into (60) implies the claimed error bound in (58).  $\square$

**Remark 8.** From the above theorem it follows that the LP feasibility problem when  $\Omega$  has no recession direction reduces to problem (P). To get an  $\epsilon_0$ -approximate solution of  $\Omega$  with a prescribed accuracy  $\epsilon_0 \leq \Delta'_0/2R'$ , from (57) it suffices to apply the Triangle Algorithm using as tolerance  $\epsilon_0\Delta'_0/2(\Delta'_0 + b_0)$ . In theory, to estimate the number of needed iterations we need to have an estimate  $\Delta'_0$ , a lower bound to  $\Delta_0$ . However, in practice given a prescribed accuracy  $\epsilon_0$  we merely need to run the Triangle Algorithm until either we have computed an  $\epsilon_0$ -approximate solution of  $\Omega$ , or a witness proving that it is empty. Despite this alternative, there is a way to get an estimate of  $\Delta_0$  as described in the next remark.

**Remark 9.** Since  $\Delta_0$  is unknown, in practice we can use an estimate. One possible approach is first to try to test if 0 lies in  $\text{conv}(\{a_1, \dots, a_n\})$ . Assuming that  $\Omega$  has no recession direction, 0 is not in this convex hull. Thus by applying the Triangle Algorithm we will get a witness  $p'$  such that  $d(p', a_i) < d(p', 0)$  for all  $i = 1, \dots, n$ . Such a point  $p'$  by Corollary 2 will necessary satisfy:

$$\frac{1}{2}d(p', 0) \leq \Delta_0 \leq d(p', 0).$$

Thus by solving this auxiliary convex hull decision problem we get a witness  $p'$  whose norm can be used as  $\Delta'_0$ . Next we use  $p'$  as the starting iterate as it already lies in the  $\text{conv}(\{a_1, \dots, a_n, -b\})$ .

Following the above we offer a two-phase Triangle Algorithm for solving the feasibility problem in LP with the assumption that  $0 \notin \text{conv}(\{a_1, \dots, a_n\})$ :

**Two-Phase Triangle Algorithm** ( $A = [a_1, \dots, a_n], b$ )

- **Phase 1.** Call **Triangle Algorithm**( $\{a_1, \dots, a_n\}, 0$ ) to get a witness  $p' \in \text{conv}(\{a_1, \dots, a_n\})$ .
- **Phase 2.** Starting with  $p'$  in Phase 1, call **Triangle Algorithm**( $\{a_1, \dots, a_n, -b\}, 0$ ).

**Theorem 10.** Suppose  $\Omega$  is nonempty and  $\text{Res}(\Omega)$  is empty. Then, given any  $\epsilon_0 \in (0, 1)$ , in order to compute an  $\epsilon_0$ -approximate solution of  $\Omega$  (i.e. a solution  $x_0 \geq 0$  such that  $d(Ax_0, b) \leq \epsilon_0 R'$ ) it suffices to set  $\Delta'_0 = 0.5d(p', 0)$ , where  $p'$  is the witness computed in Phase 1 of the Two-Phase Triangle Algorithm. Then in Phase 2 of the algorithm it suffices to compute a point  $p' \in \text{conv}(\{a_1, \dots, a_n, -b\})$  so that

$$d(p', 0) \leq \epsilon R',$$

where  $\epsilon$  satisfies

$$\epsilon \leq \frac{\Delta'_0}{2} \min \left\{ \frac{1}{R'}, \frac{\epsilon_0}{(\Delta'_0 + b_0)} \right\}.$$

In particular, since  $\Delta_0 \leq R'$ , it suffices to pick

$$\epsilon \leq \frac{\Delta'_0}{4R'} \epsilon_0.$$

Then the number of iterations in Phase 2 of the algorithm,  $K_\epsilon$ , satisfies  $K_\epsilon \leq 48\epsilon^{-2} = O(\epsilon^{-2}) = O(\epsilon_0^{-2}(R'/\Delta'_0)^2)$ .  $\square$

## 6 Solving The General LP Feasibility

Here again we consider the problem of testing if  $\Omega = \{x \in \mathbb{R}^n : Ax = b, x \geq 0\}$  is nonempty and if nonempty, computing an  $\epsilon$ -approximate solution (see (3)). Whether or not  $0 \in \text{conv}(\{a_1, \dots, a_n\})$ , it is well known that to test the feasibility of  $\Omega$  one can safely add the constraint  $\sum_{i=1}^n x_i \leq M$ , where  $M$  is a large enough constant. For integer inputs, such  $M$  can be computed, dependent on the size of encoding of  $A, b$ , see e.g. Schrijver [35]. In fact it can be shown that  $M$  can be taken to be  $O(2^{O(L)})$ , where  $L$  is the size of  $\Omega$ , dependent on  $m, n$  and logarithm of the absolute value of the largest entry of  $A$  or  $b$ , see e.g. [23].

Having such a number  $M$  at hand, by adding a slack variable,  $x_{n+1}$  to this constraint, testing the feasibility of  $\Omega$  is equivalent to testing the feasibility of

$$\Omega_M = \{(x, x_{n+1}) \in \mathbb{R}^{n+1} : Ax = b, \sum_{i=1}^{n+1} x_i = M, x \geq 0, x_{n+1} \geq 0\}. \quad (69)$$

Dividing the equations in  $\Omega_M$  by  $M$  and setting  $y_i = x_i/M$ ,  $i = 1, \dots, n+1$ , we conclude that testing the feasibility of  $\Omega_M$  is equivalent to testing the feasibility of

$$\bar{\Omega}_M = \{(y, y_{n+1}) \in \mathbb{R}^{n+1} : Ay = \frac{b}{M}, \sum_{i=1}^{n+1} y_i = 1, y \geq 0, y_{n+1} \geq 0\}. \quad (70)$$

Next, testing the feasibility of  $\bar{\Omega}_M$  is a convex hull decision problem we call the *augmented* (P): testing if  $p \in \text{conv}(\bar{S})$ , where

$$p = \frac{b}{M}, \quad \bar{S} = \{a_1, \dots, a_n, a_{n+1}\}, \quad a_{n+1} = 0. \quad (71)$$

We may solve the augmented (P) in two different ways, directly by solving a single convex hull decision problem, or by solving a sequence of such problems. We will describe the two approaches and analyze their complexities. First, we prove an auxiliary lemma, an intuitively simple geometric result.

**Lemma 2.** *Given  $u, w \in \mathbb{R}^m$ , we have:*

$$\sup \left\{ d(u, \frac{w}{\mu}) : \mu \in [1, \infty) \right\} = \max \{d(u, w), d(u, 0)\}. \quad (72)$$

*Proof.* Consider the maximum of the function  $f(t) = d^2(u, tw)$  over the interval  $t \in [0, 1]$ . Since  $f(t)$  is convex, its maximum is attained at an endpoint of the interval. Letting  $t = 1/\mu$ , the proof is complete.  $\square$

**Theorem 11.** *Suppose that a positive number  $M$  satisfies the property that  $\Omega$  is feasible if and only if  $\Omega_M$  is feasible. Then, solving the augmented (P) to within accuracy of  $\epsilon/M$  is equivalent to testing the feasibility of  $\Omega$  to within accuracy of  $\epsilon$ . More specifically, by applying the Triangle Algorithm to solve the augmented (P) (see (71)), in  $O(mnM^2\epsilon^{-2})$  arithmetic operations, either we compute a witness  $p' \in \text{conv}(\bar{S})$  proving that  $b/M \notin \bar{S}$  (hence  $\Omega = \emptyset$ ), or a point  $p' \in \text{conv}(\bar{S})$  such that for some  $i = 1, \dots, n+1$  we have,*

$$d(b, Mp') \leq \epsilon \max\{d(b, a_i), d(a_i, 0)\} \leq 2R'\epsilon, \quad (73)$$

where  $R'$  is as in (51). Equivalently,  $p' = Ay_0$ , for some  $y_0 \geq 0$ , and if  $x_0 = My_0$ , then

$$d(b, Ax_0) \leq \epsilon \max\{d(b, a_i), d(a_i, 0)\} \leq 2R'\epsilon. \quad (74)$$

*Proof.* If solving the augmented (P) to accuracy  $\epsilon/M$  leads to a witness  $p' \in \text{conv}(\bar{S})$  proving that  $b/M \notin \bar{S}$ , then  $\Omega$  is empty. Otherwise, by Theorem 8 the algorithm leads to a point  $p' \in \text{conv}(\bar{S})$  such that for some  $i = 1, \dots, n+1$  we have

$$d(\frac{b}{M}, p') \leq \frac{\epsilon}{M} d(\frac{b}{M}, a_i). \quad (75)$$

Multiplying the above by  $M$ , applying Lemma 2, and since  $Md(b/M, p') = d(b, Mp')$ , we get the proof of the first claimed inequalities in (73) and (74). The bound  $2R'\epsilon$  follows from the triangle inequality,  $d(b, a_i) \leq d(b, 0) + d(a_i, 0)$ .  $\square$

Instead of solving the augmented (P) with an a priori estimate for  $M$ , we may solve it as a sequence of augmented (P)'s with increasing estimates of  $M$  that successively doubles in value. To describe this approach, first consider the following.

Given  $\mu > 0$ , let  $p_\mu = b/\mu$ . Select any  $p' \in \text{conv}(\bar{S})$  as the iterate and let

$$\mu_0 = \sup \{ \mu : d(p_\mu, a_i) > d(p', a_i), \quad \forall i = 1, \dots, n+1 \}. \quad (76)$$

Clearly,  $0 < \mu_0 < \infty$ . According to Theorem 3, and the definition of  $\mu_0$ , for any  $\mu \in (0, \mu_0)$ ,  $p'$  is a witness proving that  $p_\mu \notin \text{conv}(\bar{S})$ . From Lemma 2, we know  $\mu_0 \geq 1$ . Given  $\epsilon > 0$ , Set  $\mu = \mu_0$  and consider the following iterative step:

**Iterative Step.** Let  $\epsilon_\mu = \epsilon/\mu$ . Use the Triangle Algorithm to solve the augmented (P) with  $p_\mu = b/\mu$  to either compute  $p'_\mu \in \text{conv}(\bar{S})$  such that

$$d(p_\mu, p'_\mu) \leq \epsilon_\mu \max\{d(p_\mu, a_i), d(a_i, 0)\}, \quad \text{for some } i,$$

or a witness  $p'_\mu \in \text{conv}(\bar{S})$  proving that  $p_\mu \notin S$ . In the latter case replace  $\mu$  with  $2\mu$  and repeat.

**Theorem 12.** *Repeating the iterative step, either we compute an  $\epsilon$ -approximate feasible point of  $\Omega$ , or a witness to the infeasibility of the augmented (P) with  $\mu \geq M$ . In the latter case  $\Omega$  is empty. More specifically, if the algorithm requires  $r$  iterations of the iterative step, its arithmetic complexity is bounded above by  $(64 \ln 2)mn\epsilon^{-2}2^{2r}R'^2 = O(mn2^{2r}\epsilon^{-2})$ . Furthermore,  $r \leq \lceil \log_2 M \rceil$ .*

*Proof.* Since  $\mu_0 \geq 1$ , the number of augmented (P)'s to be solved is bounded by  $t = \lceil \log_2 M \rceil$ . With the initial value of  $\mu = \mu_0$  we solve the augmented (P) to either compute an approximate solution in  $\Omega$  to within accuracy of  $\epsilon$ , or a witness to the infeasibility of  $\Omega_\mu$ . By Theorem 8 this takes  $O(\epsilon^{-2})$  arithmetic operations. If  $\Omega_\mu$  is infeasible, we double  $\mu$  and test the feasibility of new  $\Omega_\mu$ . Then by (32) in Lemma 1, the number of iterations needed to halve the current error that is known to exceed  $\epsilon$  is bounded by

$$(32 \ln 2) \frac{\bar{R}^2}{\epsilon^2}, \quad \bar{R} = \max\{\max\{d(b, a_i), d(a_i, 0)\} : i = 1, \dots, n+1\} \leq 2R', \quad (77)$$

where the bound  $2R'$  is from the fact that  $d(b, a_i) \leq d(b, 0) + d(a_i, 0)$ , Repeating this  $r$  times we get the total complexity bounded by

$$\bar{K}_\epsilon \leq (32 \ln 2) \frac{\bar{R}^2}{\epsilon^2} (1 + 2^2 + 2^4 + \dots + 2^{2r}) \leq (32 \ln 2) \frac{\bar{R}^2}{\epsilon^2} 2^{2r+1} = (64 \ln 2) \frac{\bar{R}^2}{\epsilon^2} 2^{2r+1} = O\left(\frac{2^{2r}}{\epsilon^2}\right).$$

□

**Concluding Remarks and Future Work.** In this article we have described several novel characterization theorems and a very simple algorithm for the convex hull decision problem, the Triangle Algorithm. The Triangle Algorithm is quite straightforward to implement and its main step is the computation of distances and their comparisons in identifying a pivot point or a witness. The simplicity of the algorithm, its theoretical properties, and the distance duality will lead to new applications. Already in [21] we give a version of the Triangle Algorithm for the case of problem (P) where we wish to locate the coordinates of a point  $p$  having prescribed distances to the sites  $S = \{v_1, \dots, v_n\}$ . We call this the *ambiguous convex hull problem* since not only the coordinates of the point are unknown, so is the existence of such point. Despite this ambiguity, a variation of the Triangle Algorithm can solve the problem in  $O(mn\epsilon^{-2} \ln \epsilon^{-1})$  arithmetic operations. We now make several comments.

(i) While the Triangle Algorithm works with distances, it requires only the four elementary operations and comparisons, no square-root operation is required. Note that the constant in the worst-case analysis in

iteration complexity in (37) is an absolute constant and is independent of the initial point  $p_0$ . This is due to the fact that the Triangle Algorithm seeks to reduce the relative error,  $d(p_k, p)/d(p, v_j)$ , a more sensible measure for the problem than seeking to reduce the absolute error,  $d(p_k, p)$ .

(ii) The iteration complexity estimate of  $O(\epsilon^{-2})$  is under the pessimistic assumption that in every iteration it has the worst-case performance, see (22), the relationship between  $\delta$  and  $\delta'$  in Theorem 7. Hence, in practice a much more efficient complexity should be expected.

(iii) In the worst-case, each iteration requires  $O(mn)$  arithmetic operations. However, in the best case, when  $p \in \text{conv}(S)$ , the algorithm can find a pivot point by searching a constant number of  $v_i$ 's, taking  $O(m)$  arithmetic operations. There are some obvious options and variations of the algorithm. For instance, given the current iterate  $p_k$ , we can use the pivot point  $v_j$  to correspond to the smallest index  $j$  satisfying  $d(p_k, v_j) \geq d(p, v_j)$ . If no such  $j$  exists,  $p_k$  is a witness to the infeasibility of  $p$ . Alternatively, we can choose among all candidate pivot points  $v_j$ , the one that minimizes the gap, the distance between the corresponding  $p''$  and  $p$ . This optimizes on reducing the gap but at the cost of more computation. This type of approach is similar to a more general version of Frank-Wolfe algorithm where all the indices are examined, see Clarkson [10], Algorithm 1.2. However, in our case we suggest using only the indices that correspond to a pivot point (i.e.  $d(p', v_j) \geq d(p, v_j)$ ). There are in fact many options for selecting a pivot, including relaxations in the computation of the closest point along the line segment  $p'v_j$ , or stronger versions where we choose a direction in the span of directions from  $p'$  to all candidate pivot points. There are many worthy experimental and theoretical considerations and will be considered in a separate article. The initial iterate  $p_0$  can be taken as any of the  $v_i$ 's, or the center of  $\text{conv}(S)$ , namely  $\sum_{i=1}^n v_i/n$ . The latter choice would of course not be a sparse approximation.

(iv) Analogous to polynomial-time algorithms for linear programming with integer inputs, given that  $p \in \text{conv}(S)$ , after  $d(p_k, p)$  has been sufficiently reduced,  $p_k$  can be rounded into an exact solution. This means the  $\alpha_i$ 's in the representation of  $p_k$  as a convex combination of  $v_i$ 's can be rounded to an exact value in representing  $p$  as a convex combination of  $v_i$ 's.

(v) Given an iterate  $p_k$ , rather than using any representation of it as a convex combination of points in  $S$  as described in (28), we can use a representation as a convex combination of at most  $m+1$  of the  $v_i$ 's. From LP theory this is possible since each point in  $\text{conv}(S)$  can be represented as such. Such representation for  $p_k$ , analogous to a basis representation in the simplex method, when  $d(p_k, p)$  is sufficiently small may allow identifying a subset of  $m+1$  points where  $p$  can be written as a convex combination of.

(vi) It may be possible to define a *path-following* version of the Triangle Algorithm for solving (P) as follows: to test if  $p \in \text{conv}(S)$ , we consider  $p + tu$ , where  $u \neq 0$  is a given auxiliary point in  $\mathbb{R}^m$ , and  $t$  is a positive real parameter, initially selected to be large enough so that  $p + tu$  is infeasible to  $\text{conv}(S)$  via a witness. The parameter  $t$  is then successively reduced to zero.

(vii) The Triangle Algorithm for (P), when applied to each  $v_i$  with  $S_i = S - \{v_i\}$  gives an approximation algorithm for solving the irredundancy problem.

(viii) The Triangle Theorem can also be considered as an algorithm that solves the intersecting open balls problem, testing if a given set of open balls in  $\mathbb{R}^m$  that are known to have a common point on their boundary, have a nonempty intersection.

Finally, the results in this article lead into new research problems and research projects, including computational testing, average case analysis, amortized complexity, generalization of the characterization theorems and the Triangle Algorithm, as well as its specialization to linear programming problems with special structures, and combinatorial and graph optimization problems. As generalizations, it is possible to state (P) and the corresponding characterizations when the set  $S$  consists of one or a finite number of compact subsets of  $\mathbb{R}^m$ . For instance, polytopes, balls, or more general convex bodies. Moreover, (P) can also be defined over other cones. For instance, a canonical problem in semidefinite programming described in [24] is an analogue of (P) over the cone of positive semidefinite matrices. We hope to investigate these in future work.

**Acknowledgements.** I thank Iraj Kalantari for a discussion on the use of Voronoi diagram argument to prove Theorem 1, resulting in a simpler geometric proof than the one given in an earlier version of this article. I also like to thank Tong Zhang for a discussion regarding the Greedy Algorithm for general convex

minimization over a simplex.

## References

- [1] T. Asano, J. Matoušek, T. Tokuyama, Zone diagrams: Existence, uniqueness, and algorithmic challenge, *SIAM Journal on Computing*, 37 (2007), 1182 - 1198.
- [2] F. Aurenhammer, Voronoi Diagrams – A survey of fundamental geometric data structure, *ACM Computing Surveys*, 23 (1991), 345 - 405.
- [3] C. Burges, A tutorial on support vector machines for pattern recognition, *Data Mining and Knowledge Discovery*, 2 (1998), 121-167.
- [4] R. Berkowitz, B. Kalantari, D. Memendendez, and I. Kalantari, On properties of forbidden zone of polyhedrons and polytopes, Proceedings of the Ninth annual International Symposium on Voronoi Diagrams in Science and Engineering, (2012), 56 - 65.
- [5] T. M. Chan, Output-sensitive results on convex hulls, extreme points, and related problems, *Discrete Comput. Geom.*, Volume 16 (1996), 369 - 387.
- [6] B. Chazelle, An optimal convex hull algorithm in any fixed dimension, *Discrete Comput. Geom.*, 10 (1993), 377 - 409
- [7] V. Chvátal, *Linear Programming*, W.H. Freeman and Company, New York, 1983.
- [8] K.L. Clarkson, Las Vegas algorithms for linear and integer programming when the dimension is small, Proceedings of 29-th Annu. IEEE Sympos. Found. Comput. Sci., (1988), 452- 456.
- [9] K.L. Clarkson, More output-sensitive geometric algorithm, Proceedings of the 35-th Annual IEEE Symposium on Foundations of Computer Science, (1994) 695 - 702.
- [10] K. L. Clarkson. Coresets, Sparse Greedy Approximation, and the Frank-Wolfe algorithm. In SODA '08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms, 922 - 931. Society for Industrial and Applied Mathematics, 2008.
- [11] S. C. de Biasi, B. Kalantari, I. Kalantari, Maximal zone diagrams and their computation, Proceedings of the Seventh Annual International Symposium on Voronoi Diagrams in Science and Engineering (2010), 171 - 180.
- [12] S. C. de Biasi, B. Kalantari, I. Kalantari, Mollified zone diagrams and their computation, *Transactions on Computational Science XIV: Special Issue on Voronoi Diagrams Delaunay Triangulation* (2011), 31 - 59.
- [13] M.E. Dyer and A.M. Frieze, A randomized algorithm for fixed-dimensional linear programming, *Math. Programming*, 44 (1989), 203 - 212.
- [14] M. Frank and P. Wolfe, An algorithm for quadratic programming, *Naval Res. Logist. Quart.*, 3 (1956), 95 - 110.
- [15] B. Gärtner and M. Jaggi, Coresets for polytope distance, Symposium on Computational Geometry (2009), 33 - 42.
- [16] E. G. Gilbert, An iterative procedure for computing the minimum of a quadratic form on a convex set, *SIAM Journal on Control* Volume 4 (1966), 61 - 80.

- [17] J. E. Goodman, J. O'Rourke (Editors), *Handbook of Discrete and Computational Geometry*, 2nd Edition (Discrete Mathematics and Its Applications) 2004, Chapman & Hall Boca Raton.
- [18] S. Har-Peled, D. Roth, and D. Zimak, Maximum margin coresets for active and noise tolerant learning. IJCAI, 2007.
- [19] Y. Jin and B. Kalantari, A procedure of Chvátal for testing feasibility in linear programming and matrix scaling, *Linear Algebra and its Applications*, 416 (2006), 795 - 798.
- [20] G. Kalai, A subexponential randomized simplex algorithm, Proceedings of the 24-th Annual ACM Symposium on Theory of Computing, (1992), 475 - 482.
- [21] B. Kalantari, Finding a lost treasure in convex hull of points from known distances. In the Proceedings of the 24th Canadian Journal of Computational Geometry (2012), 271 - 276.
- [22] B. Kalantari, Canonical problems for quadratic programming and projective methods for their solution, *Contemporary Mathematics*, 114 (1990), 243 - 263.
- [23] B. Kalantari and M.R. Emamy-K, On linear programming and matrix scaling over the algebraic numbers, *Linear Algebra and its Applications*, 262 (1997), 283-306.
- [24] B. Kalantari, Semidefinite programming and matrix scaling over the semidefinite cone, *Linear Algebra and its Applications*, 375 (2003), 221 - 243.
- [25] N. Karmarkar, A new polynomial time algorithm for linear programming, *Combinatorica*, 4 (1984), 373 - 395.
- [26] J. A. Kelner and D. A. Spielman, A randomized polynomial-time simplex algorithm for linear programming, Proceedings of the 38th Annual ACM Symposium on Theory of Computing, 2006.
- [27] L. G. Khachiyan, A polynomial algorithm in linear programming, *Doklady Akademii Nauk SSSR*, (1979), 1093 - 1096.
- [28] L. Khachiyan and B. Kalantari, Diagonal matrix scaling and linear programming, *SIAM J. Optim.*, 4 (1992), 668 - 672.
- [29] V. Klee and G.J. Minty, How good is the simplex algorithm?, In O. Shisha, editor, *Inequalities III*, 159 - 175. Academic Press, 1972.
- [30] J. Matoušek, M. Sharir, and E. Welzl, A subexponential bound for linear programming. In Proceedings of the 8th Annual ACM Symposium on Computational Geometry, (1992) 1 - 8.
- [31] J. Matoušek, Linear optimization queries, *Journal of Algorithms*, 14 (1993), 432 - 448.
- [32] N. Megiddo, Linear programming in linear time when the dimension is fixed, *Journal of the ACM*, 31 (1984), 114 - 127.
- [33] R. Motawani and R. Raghavan, *Randomized Algorithms*, Cambridge University Press, 1995.
- [34] F. P. Preparata, M. I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, New York, 1985.
- [35] A. Schrijver, *Theory of Linear and Integer Programming*, John Wiley and Sons, Inc., New York, Chichester, Brisbane, Toronto and Singapore, 1986.
- [36] R. Seidel, Small-dimensional linear programming and convex hulls made easy, *Discrete Computational Geometry*, 6 (1991), 423 - 434.



- [37] M. Sharir and E. Welzl, A combinatorial bound for linear programming and related problems. In Proceedings of the 9th Symposium on Theoretical Aspects of Computer Science, (1992), 569 - 579.
- [38] M. J. Todd, The many facets of linear programming, Math. Prog. 91 (2002), 417 - 436.
- [39] T. Zhang, Sequential greedy approximation for certain convex optimization problems, IEEE Trans. Information Theory, 49 (2003), 682 - 691.
- [40] D. A. Zimak, Algorithms and Analysis for Multi-Category Classification, Ph.D. thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, 2006.